УДК 681.3

ДВУМЕРНЫЕ ДИСКРЕТНЫЕ ПРЕОБРАЗОВАНИЯ В АЛГОРИТМАХ СЖАТИЯ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТЕЙ

Ю.Б. Буркатовская, М.А. Карагодин, А.Н. Осокин

Томский политехнический университет
E-mail: osokin@prim.ce.cctpu.edu.ru, mike@webstr.us

Рассмотрены двумерные дискретные ортогональные преобразования, которые позволяют значительно сократить время сжатия кадра видеопоследовательности и могут быть реализованы в виде аппаратного кодека.

Цифровые изображения занимают все большую часть информационного мира. Развитие сети Интернет, систем видеонаблюдения, прогресс в технологии производства цифровых камер, сканеров и принтеров привели к широкому использованию цифровых изображений. Отсюда постоянный интерес к улучшению алгоритмов сжатия данных, представляющих изображения. Сложность алгоритмов, используемых для компрессии изображений, неуклонно растет – сказанное касается не только объема вычислений, но и идейных основ построения алгоритмов. В настоящее время существуют следующие семейства алгоритмов сжатия изображений [1–3].

- Алгоритмы сжатия изображений без потерь.
- Вейвлет-сжатие (включая стандарт *JPEG*2000, основанный на дискретном вейвлет-преобразовании *DWT*).
- Фрактальное сжатие.
- Алгоритм сжатия *JPEG* и его модификации.

Сжатие изображений из чисто теоретической задачи стало повседневной практикой, что требует при его осуществлении постоянного внимания к возможностям реальной аппаратуры и экономическим характеристикам, например, для систем, обеспечивающих хранение, воспроизведение и обработку видеоинформации в мобильных сетях класса 3G, в системах видеонаблюдения, связанных с компьютером по низкоскоростной линии связи (см. научно-техническую программу «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2005–2006 гг. по приоритетному направлению «Информационно-телекоммуникационные системы» ИТ-13.4/002). Поэтому требуется разработка новых быстрых дискретных ортогональных преобразований и соответствующих алгоритмов сжатия, пригодных для практической реализации на современной элементной базе.

Определим критерии, которым должен соответствовать современный алгоритм сжатия изображений, пригодный для реализации в аппаратуре. Примем размер кадра равным 800×600, количество бит на пиксел – 24.

- Скорость работы. Для современных систем видеонаблюдения необходима скорость сжатия не менее 25 кадров/с. Это означает, что максимальное время обработки кадра должно быть равно 40 мс.
- Коэффициент сжатия. При высокой скорости работы необходимо обеспечить коэффициент сжатия изображения не ниже, чем у классического алгоритма *JPEG*, то есть, в 3–25 раз в зависимости от уровня допустимых потерь информации.
- Качество восстановленного изображения. При высокой скорости работы необходимо обеспечить качество изображения, визуально оцениваемое не ниже, чем у классического алгоритма *JPEG*, в зависимости от уровня допустимых потерь информации.
- Простота аппаратной реализации. Алгоритмы сжатия, работающие в реальном времени, должны иметь только две операции – сложение и сдвиг, чтобы обеспечить экономию аппаратных ресурсов, и, тем самым, их дешевую реализацию в аппаратном виде, например, с использованием недорогих *ПЛИС*.

После анализа существующих семейств алгоритмов сжатия изображений с позиции определенных выше критериев их оценки, авторы пришли к следующим выводам:

- Алгоритмы сжатия изображений без потерь применимы только к узким классам изображений (ограниченность цветов, большие области одного цвета и т.п.). Неэффективны для сжатия полноцветных реалистичных изображений.
- Вейвлет-алгоритмы (включая стандарт *JPEG*2000) рекурсивны, поэтому для работы требуют много времени, а также большой емкости запоминающего устройства при их аппаратной реализации. Это значительно увеличивает стоимость аппаратного кодека.
- Фрактальное сжатие основано на поиске подобных участков изображения, получаемых при помощи прямого перебора областей изображения с применением к этим участкам аффинных преобразований. Даже на современных микропроцессорных платформах программное обеспечение, реализующее фрактальные алгоритмы сжатия работает медленно. Сложность аппаратной реализации этих алгоритмов также очевидна.
- Алгоритм сжатия *JPEG* и его модификации. Скорость данных алгоритмов линейно зависима от размера изображения, они не требуют отдельного запоминающего устройства при аппаратной реализации. Также *JPEG* [1] может быть усовершенствован, и на основе модифицированного алгоритма может быть построен конкурентоспособный кодек.

Алгоритм *JPEG* использует дискретно-косинусное преобразование (*Discrete-Cosine Transforming, DCT*) [1]. Вычисление *DCT* и обратного преобразования *IDCT* является одной из наиболее трудоемких и длительных процедур, выполняемых при сжатии и распаковке кадров. Таким образом, этим процедуры в наибольшей степени нуждаются в оптимизации, позволяющей ускорить работу программного и аппаратного обеспечения, производящего сжатие и распаковку видеопоследовательностей.

Строго говоря, дискретно-косинусное преобразование не является единственно приемлемым ортогональным преобразованием для сжатия кадров; мы можем использовать и любое другое [1]. Рассмотрим подробнее этап вычисления ортогонального преобразования.

Двумерные ортогональные преобразования выполняются по следующей формуле:

$$Y = kPXP^T.$$

Здесь X: 8×8 – матрица входных значений, Y: 8×8 – матрица выходных значений (коэффициентов), P: 8×8 – матрица преобразований, k – скалярный масштабирующий коэффициент. Рассмотрим теперь матрицу Y^T:

$$Y^T = (kPXP^T) = kPX^T P^T.$$

Введем обозначение $Z=PX^T$. Тогда

$$Y^T = kZP^T;$$

$$Y = (kZP^T)^T = kPZ^T.$$

Таким образом, вычисление матрицы выходных значений Y может быть сведено к двум последовательно выполняемым операциям перемножения двух матриц, в которых первая матрица одинаковая. Это упрощает программную и аппаратную реализацию алгоритма.

Рассмотрим теперь операцию перемножения двух матриц 8×8. Вычисление одного столбца результирующей матрицы в общем случае требует 64 умножений и 56 сложений. Наши дальнейшие рассуждения будут направлены на то, чтобы уменьшить число этих операций для некоторых частных случаев ортогональных преобразований (матриц P). Это достигается путем разложения матрицы преобразования P на разреженные множители, т.е. представления ее в виде

$$P = P_1 \times P_2 \times \ldots \times P_m, \quad (1)$$

где матрицы P_i содержат максимально возможное количество нулей. Такой подход применяется, в частности, в [1].

Еще одним методом оптимизации двумерного ортогонального преобразования является аппроксимация. Задача аппроксимации сводится к приведению коэффициентов матрицы к виду $m/2^n$. Это позволяет избавиться от длительных операций с плавающей точкой, а также от ресурсоемкой операции умножения, заменив ее сложениями и сдвигами.

Матрица P для вычисления дискретно-косинусного преобразования выглядит следующим образом [1]:

$$P = \begin{bmatrix} 0,3535 & 0,3535 & 0,3535 & 0,3535 & 0,3535 & 0,3535 & 0,3535 & 0,3535 \\ 0,4904 & 0,4157 & 0,2778 & 0,0975 & -0,0975 & -0,2778 & -0,4157 & -0,4904 \\ 0,4619 & 0,1913 & -0,1913 & -0,4619 & -0,4619 & -0,1913 & 0,1913 & 0,4619 \\ 0,4157 & -0,0975 & -0,4904 & -0,2778 & 0,2778 & 0,4904 & 0,0975 & -0,4157 \\ 0,3535 & -0,3535 & -0,3535 & 0,3535 & 0,3535 & -0,3535 & -0,3535 & 0,3535 \\ 0,2778 & -0,4904 & 0,0975 & 0,4157 & -0,4157 & -0,0975 & 0,4904 & -0,2778 \\ 0,1913 & -0,4619 & 0,4619 & -0,1913 & -0,1913 & 0,4619 & -0,4619 & 0,1913 \\ 0,0975 & -0,2778 & 0,4157 & -0,4904 & 0,4904 & -0,4157 & 0,2778 & -0,0975 \end{bmatrix}.$$

Проведем аппроксимацию преобразования с точностью до $^1/_{16}$. Для этого умножим каждый элемент матрицы на 16, разделим его на $\sqrt{1/8}$=0,3535 и округлим до четного числа, так, чтобы минимизировать количество получившихся значений. Выбор числа 16 обусловлен аппаратными ограничениями. Дело в том, что для сохранения элемента выходной матрицы необходимо 12 бит. Типичная разрядность представления таких данных в архитектуре i386 – 16 бит, то есть, для проведения 16-битовых операций необходимо умножение каждого элемента входной матрицы на 2^4=16.

После таких преобразований получим матрицу P':

$$P' = \begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 22 & 18 & 12 & 4 & -4 & -12 & -18 & -22 \\ 22 & 8 & -8 & -22 & -22 & -8 & 8 & 22 \\ 18 & -4 & -22 & -12 & 12 & 22 & 4 & -18 \\ 16 & -16 & -16 & 16 & 16 & -16 & -16 & 16 \\ 12 & -22 & 4 & 18 & -18 & -4 & 22 & -12 \\ 8 & -22 & 22 & -8 & -8 & 22 & -22 & 8 \\ 4 & -12 & 18 & -22 & 22 & -18 & 12 & -4 \end{bmatrix}.$$

Заметим, что каждая строка матрицы P' может быть представлена в одной из двух следующих форм:

$$[a\ b\ c\ d\ d\ c\ b\ a],$$
$$[a\ b\ c\ d\ \text{-}d\ \text{-}c\ \text{-}b\ \text{-}a]. \qquad (2)$$

Таким образом, для вычисления произведения $P'\times T$, где $T=[t_0,...,t_7]^T$ – вектор-столбец, требуются выражения $t_0–t_7,...,t_4+t_5,t_4–t_5$; т.е., матрица P_m в разложении (1) имеет вид:

$$P_m = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}. \qquad (3)$$

Умножение этой матрицы на вектор T требует 8 сложений. В итоге получится вектор:

$$P_m T = T' = \begin{bmatrix} t_0+t_7 \\ t_0-t_7 \\ t_1+t_6 \\ t_1-t_6 \\ t_2+t_5 \\ t_2-t_5 \\ t_3+t_4 \\ t_3-t_4 \end{bmatrix}. \qquad (4)$$

Рассмотрим строки, представимые в виде (2). Произведение их на вектор T имеет вид:

$$[a\ \ b\ \ c\ \ d\ \ d\ \ c\ \ b\ \ a]\times T =$$
$$=[a\ \ 0\ \ b\ \ 0\ \ c\ \ 0\ \ d\ \ 0]\times T',$$
$$[a\ \ b\ \ c\ \ d\ \ \text{-}d\ \ \text{-}c\ \ \text{-}b\ \ \text{-}a]\times T =$$
$$=[0\ \ a\ \ 0\ \ b\ \ 0\ \ c\ \ 0\ \ d]\times T',$$

и требует 4 умножения и 3 сложения (если уже вычислен вектор T'). Таким образом, используя (3) и (4), получаем разложение матрицы P' на разреженные множители:

$$P' = P_1 \times P_2 = \begin{bmatrix} 16 & 0 & 16 & 0 & 16 & 0 & 16 & 0 \\ 0 & 22 & 0 & 18 & 0 & 12 & 0 & 4 \\ 22 & 0 & 8 & 0 & -8 & 0 & -22 & 0 \\ 0 & 18 & 0 & -4 & 0 & -22 & 0 & -12 \\ 16 & 0 & -16 & 0 & -16 & 0 & 16 & 0 \\ 0 & 12 & 0 & -22 & 0 & 4 & 0 & 18 \\ 8 & 0 & -22 & 0 & 22 & 0 & -8 & 0 \\ 0 & 4 & 0 & -12 & 0 & 18 & 0 & -22 \end{bmatrix} \times$$
$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}.$$

Теперь для подсчета $P'\times T$ требуется 32 умножения и 32 сложения, то есть трудоемкость алгоритма снижена в 2 раза. Правильность разложения может быть проверена перемножением полученных матриц. Строки 0, 2, 4, 6 матрицы P_1 могут быть представлены в одной из следующих форм:

$$[a\ \ 0\ \ b\ \ 0\ \ b\ \ 0\ \ a\ \ 0],$$
$$[a\ \ 0\ \ b\ \ 0\ \ \text{-}b\ \ 0\ \ \text{-}a\ \ 0].$$

Используя те же идеи, и опуская промежуточные выкладки, получаем:

$$P' = \begin{bmatrix} 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 22 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & -22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times$$
$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times$$
$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 22 & 0 & 18 & 0 & 12 & 0 & 4 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 18 & 0 & -4 & 0 & -22 & 0 & -12 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 12 & 0 & -22 & 0 & 4 & 0 & 18 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 4 & 0 & -12 & 0 & 18 & 0 & -22 \end{bmatrix} \times$$
$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}.$$

Таким образом, получение вектора$P'\times T$ потребует 22 умножения и 16 сложений. Как можно заметить, в качестве множителей для операции умножения используется всего шесть чисел: 4, 8, 12, 16, 18 и 22. Приведем операции умножения коэффициента a на каждый из этих множителей к операциям сложения и сдвига.

Исходя из данных табл. 1, 22 операции умножения можно заменить на 14 операций сдвига и 10 операций сложения. Итого, для получения век-

тора $P'\times T$ необходимо провести 14 операций сдвига и 30 операций сложения. То есть, для вычисления преобразования для одной матрицы 8×8 необходимо 2·8·14 = 224 операции сдвига и 2·8·30=480 операций сложения.

Таблица 1. *Замена операций умножения на операции сдвига и сложения*

Операция	Сдвигов	Сложений
$4a=a<<2=b_0$	1	–
$8a=a<<3=b_1$	1	–
$12a=a<<3+a<<2=b_0+b_1=b_2$	–	1
$16a=a<<4=b_3$	1	–
$18a=16a+2a=b_3+a<<1=b_4$	1	1
$22a=18a+4a=b_4+b_0=b_5$	–	1

Дискретно-косинусное преобразование не является единственным ортогональным преобразованием, которое возможно применить для сжатия изображений. Преобразование Уолша [2] характерно тем, что его возможно выполнить при помощи одних лишь операций сложения и вычитания. Однако существует способ дальнейшей оптимизации данного преобразования.

Поставим задачу разложения матрицы преобразования Уолша H_h на разреженные множители. Для этого сначала рассмотрим общий случай – блочную матрицу $T=\begin{bmatrix} Q & Q \\ Q & -Q \end{bmatrix}$, где Q – произвольная матрица размерности $m\times m$. Разложение матрицы T на разреженные множители имеет вид:

$$T=\begin{bmatrix} Q & Q \\ Q & -Q \end{bmatrix}=\begin{bmatrix} E_m & E_m \\ E_m & -E_m \end{bmatrix}\begin{bmatrix} Q & O_m \\ O_m & Q \end{bmatrix},$$

где E_m – единичная матрица размерности $m\times m$, O_m – нулевая матрица размерности $m\times m$.

Действительно, пусть $X=(x_1,...,x_{2m})^T$ – вектор размерности $2m$, $X'=(x_1,...,x_m)^T$, $X''=(x_{m+1},...,x_{2m})^T$, тогда:

$$TX=\begin{bmatrix} E_m & E_m \\ E_m & -E_m \end{bmatrix}\begin{bmatrix} Q & O_m \\ O_m & Q \end{bmatrix}\begin{bmatrix} X' \\ X'' \end{bmatrix}=$$

$$=\begin{bmatrix} E_m & E_m \\ E_m & -E_m \end{bmatrix}\begin{bmatrix} QX' \\ QX'' \end{bmatrix}=\begin{bmatrix} QX'+QX'' \\ QX'-QX'' \end{bmatrix}.$$

С другой стороны,

$$TX=\begin{bmatrix} Q & Q \\ Q & -Q \end{bmatrix}\begin{bmatrix} X' \\ X'' \end{bmatrix}=\begin{bmatrix} QX'+QX'' \\ QX'-QX'' \end{bmatrix}. \quad (5)$$

Таким образом, разложение верно. Вернемся теперь к матрице H_h. Для разложения ее на разреженные множители, уточним ее структуру. Введем матрицы:

$$A=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\quad B=\begin{bmatrix} A & A \\ A & -A \end{bmatrix}. \quad (6)$$

С использованием этих обозначений матрицу H_h можно записать:

$$H_h=\begin{bmatrix} B & B \\ B & -B \end{bmatrix}.$$

Используя (5) и (6) получаем:

$$H_h=\begin{bmatrix} E_4 & E_4 \\ E_4 & -E_4 \end{bmatrix}\begin{bmatrix} B & O_4 \\ O_4 & B \end{bmatrix};$$

$$B=\begin{bmatrix} E_2 & E_2 \\ E_2 & -E_2 \end{bmatrix}\begin{bmatrix} A & O_2 \\ O_2 & A \end{bmatrix}.$$

Таким образом, матрица H_h может быть представлена как произведение трех разреженных множителей:

$$H_h=\begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0&0 \\ 0&0&1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1 \\ 1&0&0&0&-1&0&0&0 \\ 0&1&0&0&0&-1&0&0 \\ 0&0&1&0&0&0&-1&0 \\ 0&0&0&1&0&0&0&-1 \end{bmatrix}\times$$

$$\times\begin{bmatrix} 1&0&1&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0 \\ 1&0&-1&0&0&0&0&0 \\ 0&1&0&-1&0&0&0&0 \\ 0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&1&0&-1&0 \\ 0&0&0&0&0&1&0&-1 \end{bmatrix}\times$$

$$\times\begin{bmatrix} 1&1&0&0&0&0&0&0 \\ 1&-1&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&1&-1&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&1&-1&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&1&-1 \end{bmatrix}.$$

Для умножения матрицы преобразования на один вектор-столбец исходных данных необходимо всего 24 сложения. Это означает, что для преобразования всей матрицы 8×8 необходимо 2·8·24=384 сложения, т.к. всего вектор-столбцов в матрице исходных данных восемь, и для вычисления двумерного преобразования матричные умножения на разреженные множители необходимо производить дважды – для вектор-строк и для вектор-столбцов.

Тестирование разработанного алгоритма проводилось на компьютере Pentium IV – 1,7 GHz / 256 Mb DDR RAM / 120 Gb HDD, chipset Intel i845D под управлением ОС Microsoft Windows XP (Service Pack 2). Ис-

Таблица 2. *Результаты тестирования алгоритмов, построенных на основе быстрого и классического преобразования Уолша*

Имя файла	Размер файла исходный, байт	Размер сжатого файла, байт	$K_{сж}$	Быстрое двумерное преобразование Уолша				Классическое преобразование Уолша			
				$T_{сж}$, мс	$T_{расп}$, мс	Скорость сжатия, кб/с	Скорость расп-ки, кб/с	$T_{сж}$, мс	$T_{расп}$, мс	Скорость сжатия, кб/с	Скорость расп-ки, кб/с
clegg.bmp	2148978	756439	2,84	1390	781	1546,03	2751,57	1610	1344	1334,77	1598,94
frymire.bmp	3706188	1192680	3,11	1859	1406	1993,65	2635,98	2828	2328	1310,53	1592,01
lena.bmp	786450	222854	3,53	391	313	2011,38	2512,62	625	531	1258,32	1481,07
monarch.bmp	1179666	288107	4,09	672	656	1755,46	1798,27	859	735	1373,30	1604,99
peppers.bmp	786450	196604	4,00	500	500	1572,90	1572,90	547	640	1437,75	1228,83
sail.bmp	1179666	401513	2,94	1094	531	1078,31	2221,59	1313	812	898,45	1452,79
serrano.bmp	1498296	415437	3,61	953	578	1572,19	2592,21	1047	1094	1431,04	1369,56
tulips.bmp	1179666	411649	2,87	875	500	1348,19	2359,33	969	797	1217,41	1480,13
Итого:	12465360	3885283	3,21	7734	5265	1611,76	2367,59	9798	8281	1272,24	1505,30

Примечание: $K_{сж}$ – коэффициент сжатия, $T_{сж}$, $T_{расп}$ – время сжатия и распаковки

Таблица 3. *Результаты тестирования алгоритмов, построенных на основе аппроксимированного и классического преобразования DCT*

Имя файла	Размер файла исходный, байт	Аппроксимированное DCT					Классическое DCT				
		$K_{сж}$	$T_{сж}$, мс	$T_{расп}$, мс	Скорость сжатия, кб/с	Скорость расп-ки, кб/с	$K_{сж}$	$T_{сж}$, мс	$T_{расп}$, мс	Скорость сжатия, кб/с	Скорость расп-ки, кб/с
clegg.bmp	2148978	2,64	1441	954	1491,20	2251,55	2,16	2594	1718	828,44	1250,86
frymire.bmp	3706188	3,07	2422	1675	1530,08	2212,65	2,51	4360	3015	850,04	1229,25
lena.bmp	786450	3,37	495	373	1588,79	2106,56	2,76	891	672	882,66	1170,31
monarch.bmp	1179666	4,41	712	556	1657,61	2123,40	3,62	1281	1000	920,89	1179,67
peppers.bmp	786450	4,12	503	391	1562,48	2013,67	3,38	906	703	868,05	1118,71
sail.bmp	1179666	3,06	929	556	1269,98	2123,40	2,51	1672	1000	705,54	1179,67
serrano.bmp	1498296	3,49	894	677	1675,11	2214,23	2,86	1610	1218	930,62	1230,13
tulips.bmp	1179666	2,86	990	521	1191,58	2266,17	2,35	1782	937	661,99	1258,98
Итого:	12465360	3,16	8387	5702	1486,33	2186,27	2,59	15096	10263	825,74	1214,59

пользовался международный стандартный тестовый набор *Waterloo Bragzone* (*Calgary Corpus*) [3] и специально разработанный программно-экспериментальный комплекс [4]. Результаты тестирования алгоритмов сжатия на основе рассмотренных ортогональных преобразований приведены в табл. 2 и 3.

Разработанный метод вычисления преобразования Уолша при том же качестве восстановленного изображения превосходит классический метод вычисления дискретно-косинусного преобразования по скорости сжатия и распаковки в 1,9 раза.

Разработанный аппроксимированный метод вычисления дискретно-косинусного преобразования при том же качестве восстановленного изображения превосходит классический метод вычисления дискретно-косинусного преобразования по скорости сжатия и распаковки в 1,8 раза. Сравнительная трудоемкость преобразований приведена в табл. 4.

Коэффициент сжатия для обоих преобразований возрастает примерно на 22 % по сравнению с классическим дискретно-косинусным преобразованием.

Итак, авторами разработаны и протестированы упрощенный метод вычисления дискретно-косинусного преобразования и быстрое двумерное преобразование Уолша. Оба они превосходят классический метод DCT более чем в два раза и по сжатию, и по распаковке, при том же, или более высоком коэффициенте сжатия и высоком качестве восстановленного изображения.

Таблица 4. *Трудоемкость преобразований (число операций для матрицы 8x8)*

Преобразование	Сложений	Сдвигов	Умножений
Классическое DCT	896	–	1024
Аппроксимированное DCT	480	224	–
Классическое преобразование Уолша	896	–	–
Быстрое двумерное преобразование Уолша	384	–	–

Для оценки качества изображения используется такая величина, как среднеквадратичное отклонение восстановленного изображения от исходного.

Рис. 1. *Качество восстановленного изображения*

На рис. 1 приведен график зависимости среднеквадратичного отклонения от коэффициента сжатия. На рис. 2–5 можно увидеть исходное изо-

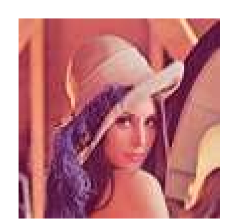


Рис. 2. *Исходное изображение*

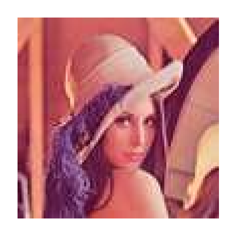


Рис. 3. *Изображение, восстановленное после сжатия с использованием классического DCT при $K_{сж}$=2,76*

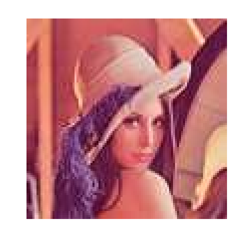


Рис. 4. *Изображение, восстановленное после сжатия с использованием аппроксимированного DCT при $K_{сж}$=3,37*

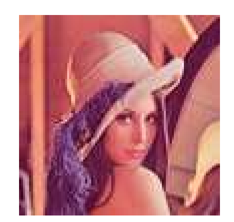


Рис. 5. *Изображение, восстановленное после сжатия с использованием быстрого преобразования Уолша при $K_{сж}$=3,53*

бражение и результаты работы алгоритмов сжатия и распаковки с использованием рассмотренных выше преобразований.

Упрощенный метод вычисления дискретно-косинусного преобразования совместим с любым другим методом сжатия изображений, имеющим в своей основе дискретно-косинусное преобразование (*JPEG, MPEG*), поэтому его очень удобно использовать для быстрой распаковки уже имеющихся сжатых изображений или видеопоследовательностей. Поскольку данное преобразование фактически является разновидностью *DCT*, средний коэффициент сжатия алгоритма на базе этого преобразования сравним с коэффициентом сжатия алгоритма *JPEG*. Рекомендуемые сферы применения: программные и недорогие аппаратные кодеки (например, в цифровых фотоаппаратах) для записи, хранения и просмотра видеопоследовательностей в широкораспространенных форматах.

Быстрое двумерное преобразование Уолша имеет низкие требования к аппаратуре и высокую скорость работы при достаточно высоком качестве восстановленного изображения, хотя и несколько уступающем алгоритму *JPEG*. Это делает данное преобразование эффективным применительно к программным и аппаратным системам видеонаблюдения, видеосвязи и видеоконференций, системам терминального доступа, а также ко всем приложениям, требующим быстрой передачи видеоданных.

СПИСОК ЛИТЕРАТУРЫ

1. Миано Дж. Форматы и алгоритмы сжатия изображений в действии. – М.: Триумф, 2003. – 336 с.
2. Карагодин М.А., Осокин А.Н. Сжатие полноцветных изображений с использованием функций Уолша // Цифровая обработка сигналов и ее применение: Докл. 6-й Междунар. конф. – М., 2004. – Т. 2. – С. 143–146.
3. Сайт Waterloo BragZone (Calgary Corpus): http://links.uwaterloo.ca/bragzone.base.html
4. Karagodin M.A. Program experimental complex for the development of the image compression algorithm // Modern Techniques and Technologies: Proc. of the 9th Intern. Scientific and Practical Conf. of Students, Post-graduates and Young Scientists. – Tomsk: Tomsk Polytechnic University, 2003. – P. 220–222.